PoseIRM: Enhance 3D Human Pose Estimation on Unseen Camera Settings via Invariant Risk Minimization

Supplementary Material

This appendix can be divided into three parts. Precisely,

1. Section A presents a detailed derivation of the Minimum Distance D_{min}.
2. Section B provides the formal expression of the loss function about 3D pose and camera parameter.
3. Section C gives more implementation details about our experiments, models and training.

A. Minimum Distance D_{min} Calculation

Note that, in the test time, the task of HPE is to estimate the locations of the keypoints of a person **within the image**, i.e., camera's screen. Therefore, to make the training and test data consistent, we need to ensure the all the keypoints of the generated 2D poses also lie in the image and thus we need to calculate the minimum distance D_{min} (Section 3.1). Below, we give the detailed steps to derive the expression of D_{min}.

We adopt the image coordinate system to determine whether a point falls within the camera's screen. We first normalize the coordinates of an image into the square $[-1, 1] \times [-1, 1]$. Let $(\mathcal{J}_u, \mathcal{J}_v) \in \mathbb{R}^2$ be the coordinates of a keypoint in a pose. Therefore, to make the keypoints located in the image, it is equivalent to ensure that,

$$\mathcal{J}_u \in [-1, 1] \text{ and } \mathcal{J}_v \in [-1, 1]. \tag{11}$$

Below, we try to derive the minimal distance to ensure the above constraints always hold.

We let $(J_x^{(w)}, J_y^{(w)}, J_z^{(w)}) \in \mathbb{R}^3$ to be a point in three-dimensional space under the world coordinate system. Then, its projection $(\mathcal{J}_u, \mathcal{J}_v)$ on the camera plane can be presented as:

$$\begin{bmatrix} \mathcal{J}_u \\ \mathcal{J}_v \\ 1 \end{bmatrix} = \lambda_s \cdot \begin{bmatrix} \mathcal{F}_x & 0 & \mathcal{C}_x \\ 0 & \mathcal{F}_y & \mathcal{C}_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{R}_{11} & \cdots & \mathcal{R}_{13} & \mathcal{T}_x \\ \vdots & \ddots & \vdots & \mathcal{T}_y \\ \mathcal{R}_{31} & \cdots & \mathcal{R}_{33} & \mathcal{T}_z \end{bmatrix} \begin{bmatrix} \mathcal{J}_x^{(w)} \\ \mathcal{J}_y^{(w)} \\ \mathcal{J}_z^{(w)} \\ 1 \end{bmatrix}, \tag{12}$$

where the parameters $\mathcal{F}, \mathcal{C}, \mathcal{T}$ and $\mathcal{R}$ are defined in the main text, and $\lambda_s \in \mathbb{R}$ is a normalization factor ensuring that the third component of the transformed vector $[\mathcal{J}_u, \mathcal{J}_v, 1]^T$ is 1. Here, for the convenience of computation, we use the matrix representation $\mathcal{R}$ for camera orientation, which has the freedom of 3. We denote the keypoints rotated in camera view as $(\mathcal{J}_x, \mathcal{J}_y, \mathcal{J}_z) \in \mathbb{R}^3$, that is

$$\begin{bmatrix} \mathcal{J}_x \\ \mathcal{J}_y \\ \mathcal{J}_z \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{11} & \cdots & \mathcal{R}_{13} \\ \vdots & \ddots & \vdots \\ \mathcal{R}_{31} & \cdots & \mathcal{R}_{33} \end{bmatrix} \begin{bmatrix} \mathcal{J}_x^{(w)} \\ \mathcal{J}_y^{(w)} \\ \mathcal{J}_z^{(w)} \end{bmatrix}. \tag{13}$$

From Equations 12 and 13, it follows that:

$$\begin{cases} \lambda_s = \dfrac{1}{\mathcal{J}_z + \mathcal{T}_z} \\ \mathcal{J}_u = \dfrac{\mathcal{F}_x(\mathcal{J}_x + \mathcal{T}_x) + \mathcal{C}_x(\mathcal{J}_z + \mathcal{T}_z)}{\mathcal{J}_z + \mathcal{T}_z} \\ \mathcal{J}_v = \dfrac{\mathcal{F}_y(\mathcal{J}_y + \mathcal{T}_y) + \mathcal{C}_y(\mathcal{J}_z + \mathcal{T}_z)}{\mathcal{J}_z + \mathcal{T}_z} \end{cases}. \tag{14}$$

Notice that, in reality, the camera center $\mathcal{C}$, is very close to the center of the image, that is,

$$\mathcal{C}_x \approx 0 \text{ and } \mathcal{C}_y \approx 0. \tag{15}$$

Moreover, $\mathcal{T}_x$, and $\mathcal{T}_y$ are significantly smaller than the coordinates of the peripheral keypoints $(\mathcal{J}_x, \mathcal{J}_y)$, i.e.,

$$|\mathcal{T}_x| \ll |\mathcal{J}_x| \text{ and } |\mathcal{T}_y| \ll |\mathcal{J}_y|. \tag{16}$$

Thus, by combining the Equations 15 and 16, Equations 14 can be approximated as:

$$\begin{cases} \mathcal{J}_u \approx \dfrac{\mathcal{F}_x \mathcal{J}_x}{\mathcal{J}_z + \mathcal{T}_z} \\ \mathcal{J}_v \approx \dfrac{\mathcal{F}_y \mathcal{J}_y}{\mathcal{J}_z + \mathcal{T}_z} \end{cases}. \tag{17}$$

By substituting Equation 17 into Equation 11 and using the fact that $\mathcal{T}_z + \mathcal{J}_z > 0$, we obtain:

$$\mathcal{T}_z \geq \max(|\mathcal{F}_x \mathcal{J}_x|, |\mathcal{F}_y \mathcal{J}_y|) - \mathcal{J}_z. \tag{18}$$

Thus, we can get the minimal distance between camera viewpoint and look-at point:

$$\boldsymbol{D_{min} = \max_{\mathcal{J}^i \in \mathcal{J}} \left(\max(|\mathcal{F}_x \mathcal{J}_x^i|, |\mathcal{F}_y \mathcal{J}_y^i|) - \mathcal{J}_z^i\right)}, \tag{19}$$

where $\mathcal{J}$ is all the keypoints in the given dataset. (There is a typo in the expression of D_{min} in the maintext.)

Notably, owing to the presence of camera distortion and the approximation error in the derivation, when directly using D_{min} as the distance between camera viewpoint and look-at point, some projected keypoints near the screen edges might fall outside due to distortion effects. To mitigate this, we sample a slighter lagrer distance D as follows:

$$D \sim \mathcal{U}(\frac{1}{1-\lambda_{scale}}D_{min}, \frac{1}{1-\lambda_{scale}}D_{min} + \lambda_D), \quad (20)$$

which is mentioned in the main text and $\mathcal{U}(a, b)$ is the uniform distribution over the interval $[a, b]$.

This adjustment helps in accommodating the distortion.

B. 3D Pose and Camera Parameter Errors

Let us first define the network of our PoseIRM as follows.

We formally define the backbone (i.e., the single-view feature extractor and the multi-view feature fusion) as

$$f_\Phi(\cdot) : \mathbb{R}^{V\times T\times J\times 2} \to \mathbb{R}^{V\times T\times C}, \quad (21)$$

where the backbone with the parameters Φ takes $V \times T$ 2D poses of J joints from V viewpoints and T frames as input and output the feature in $\mathbb{R}^{V\times T\times C}$.

The 3d pose regression head takes the above extracted feature to regress 3D poses of V viewpoints and it takes the form of

$$g_{\boldsymbol{v}_p^e}(\cdot) : \mathbb{R}^{V\times T\times C} \to \mathbb{R}^{V\times J\times 3}, \quad (22)$$

where $\boldsymbol{v}_p^e$ is the parameter vector of this pose regression head.

The camera parameter regression head with the parameters $\boldsymbol{v}_c^e$ predicts the camera parameters of V viewpoints. The camera parameters involve the camera orientation $\mathcal{R} \in \mathbb{R}^3$ in rotation vector representation, translation $\mathcal{T} \in \mathbb{R}^3$, focal length $\mathcal{F} \in \mathbb{R}^2$, center $\mathcal{C} \in \mathbb{R}^2$, and distortion $\mathcal{D} = (k_1, k_2, k_3, p_1, p_2)^\top \in \mathbb{R}^5$. Thus, the camera parameter regression head can be represented as

$$h_{\boldsymbol{v}_c^e}(\cdot) : \mathbb{R}^{V\times T\times C} \to \mathbb{R}^{V\times(3+3+2+2+5)}. \quad (23)$$

We group the parameters $\boldsymbol{v}_p^e, \boldsymbol{v}_c^e$ and Φ together into ω, i.e., $\omega = \{\boldsymbol{v}_p^e, \boldsymbol{v}_c^e, \Phi\}$ and define $(\hat{R}, \hat{T}, \hat{F}, \hat{C}, \hat{D})$ to be the output of h, i.e.,

$$(\hat{\mathcal{R}}, \hat{\mathcal{T}}, \hat{\mathcal{F}}, \hat{\mathcal{C}}, \hat{\mathcal{D}}) = h(f_\Phi(\mathbf{x}^e), \boldsymbol{v}_c^e), \quad (24)$$

where $\mathbf{x}^e \in \mathbb{R}^{V\times T\times J\times 2}$ is the input, $\hat{\mathcal{R}} \in \mathbb{R}^3$ is the predicted rotation vector, $\hat{\mathcal{T}} \in \mathbb{R}^3$ is the predicted translation vector, $\hat{\mathcal{F}} \in \mathbb{R}^2$ is the predicted focal length, $\hat{\mathcal{C}} \in \mathbb{R}^2$ is the predicted camera center, and $\hat{\mathcal{D}} \in \mathbb{R}^5$ is the predicted camera distortion.

Subsequently, we construct a loss for each output to facilitate supervised learning. For 3D poses, we apply standard MPJPE as the loss. For camera orientation $\mathcal{R}$, we apply cosine similarity loss to the rotation axis $\frac{\mathcal{R}}{\|\mathcal{R}\|_2}$ and ℓ_1 loss to the rotation angle $\|\mathcal{R}\|_2$. For other terms, i.e., $\mathcal{T}, \mathcal{F}, \mathcal{C},$ and $\mathcal{D}$, we adopt ℓ_1 loss.

Finally, suppose we are given the n_e samples $\{(\mathbf{x}_i^e, \mathbf{y}_{p,i}^e, \mathbf{y}_{c,i}^e)\}_{i=1}^{n_e}$ in the environment e, where $\mathbf{x}_i^e \in \mathbb{R}^{V\times T\times J\times 2}$ is the input 2D poses, $\mathbf{y}_{p,i}^e \in \mathbb{R}^{V\times J\times 3}$ is the ground truth 3D poses, and $\mathbf{y}_{c,i}^e \in \mathbb{R}^{V\times(3+3+2+2+5)}$ is the ground truth camera parameters. We define the risk $\mathcal{R}^e(\omega)$ (known as the loss in deep learning community) of the environment e to be the sum of 3D pose loss $\mathcal{L}_{MPJPE}(\boldsymbol{v}_p^e, \Phi)$ and camera parameter loss $\mathcal{L}_{Cam}(\boldsymbol{v}_p^e, \Phi)$. That is,

$$\mathcal{R}^e(\omega) = \mathcal{L}_{MPJPE}(\omega) + \lambda_c \mathcal{L}_{Cam}(\omega), \quad (25)$$

where

$$\mathcal{L}_{MPJPE}(\omega) = \frac{1}{n_e}\sum_{i=1}^{n_e} MPJPE(g_{\boldsymbol{v}_p^e}(f_\Phi(\mathbf{x}_i^e)), \mathbf{y}_{p,i}^e) \quad (26)$$

and,

$$\mathcal{L}_{Cam}(\omega) = \frac{1}{n_e}\sum_{i=1}^{n_e} \ell_c(h_{\boldsymbol{v}_c^e}(f_\Phi(\mathbf{x}_i^e)), \mathbf{y}_{c,i}^e) \quad (27)$$

and,

$$\begin{aligned} \ell_c(\hat{\mathbf{y}}_c, \mathbf{y}_c) = \ell_R(\hat{\mathcal{R}}, \mathcal{R}) + \lambda_T \ell_1(\hat{\mathcal{T}}, \mathcal{T}) \quad &+ \lambda_F \ell_1(\hat{\mathcal{F}}, \mathcal{F}) \\ + \lambda_C \ell_1(\hat{\mathcal{C}}, \mathcal{C}) \quad &+ \lambda_D \ell_1(\hat{\mathcal{D}}, \mathcal{D}) \end{aligned} \quad (28)$$

where ℓ_1 is the ℓ_1 loss, and ℓ_R is the loss of camera orientation taking the form of

$$\ell_R(\hat{\mathcal{R}}, \mathcal{R}) = \ell_{cosine}(\frac{\hat{\mathcal{R}}}{\|\hat{\mathcal{R}}\|_2}, \frac{\mathcal{R}}{\|\mathcal{R}\|_2}) + \ell_1(\|\hat{\mathcal{R}}\|_2, \|\mathcal{R}\|_2), \quad (29)$$

where ℓ_{cosine} is the cosine similarity loss.

C. Implementation Details

We implement our PoseIRM with Pytorch. Four Nvidia RTX 3090 GPUs are used for training and testing. We train our model with Adam [13] optimizer for 100 epochs with weight decay of 0.05. The learning rate is initially set to 2e-4 and is decreased by a factor of 0.1 for every 50 epochs. The batch size is set to 400. We follow the model setting of PoseFormer, that is, the number of input frames T and views V are 27 and 4, respectively, while the channel of joint feature C_J and pose feature C are 32 and 544, separately. The scaling factor λ_{scale} and λ_D are 0.05 and 1, separately. The weight λ of IRM regular $\mathcal{J}(\omega)$ is set to 100 in the main experiment. The weight of losses $\lambda_c, \lambda_T, \lambda_F, \lambda_C$ and λ_D are set to 0.1,0.1,1,1, and 1, respectively.